NOKIA

How to develop Java™ Applications for the Nokia 9210 Communicator

Version 1.1
09-08-01

Table of contents

1. INTRODUCTION 4

2. JAVA IMPLEMENTATION ON THE NOKIA 9210 COMMUNICATOR 4

3. DEVELOPMENT GUIDELINES 5

3.1 USER INTERFACE PRINCIPLES 5
3.2 ABSTRACT WINDOW TOOLKIT (AWT) INTERFACES 6
3.3 COMMAND BUTTON AREA API 6
3.4 STATUS PANE API 7
3.5 VIRTUAL CURSOR API 7

4. SHORT CODE EXAMPLE 7

4.1 THE SOURCE CODE 9
4.2 SYMBIAN PLATFORM SPECIFIC PROCESSES 9
4.2.1 Making an installation package for the target device 14

5. FURTHER INFORMATION 15

Change history

09-08-01	Version 1.1	Document updated by Forum Nokia
03-01-01	Version 1.0	Document created by Forum Nokia

Copyright © 2001. Nokia Mobile phones. All rights reserved.

Disclaimer:

The software or the copyrighted document is provided "as is", without warranties of any kind, and specifically excluding all implied warranties, for example of merchantability, fitness for purpose, title and non-infringement. In no event shall nokia be liable for any special, indirect or consequential damages, or any damages whatsoever resulting from loss of use, data or profits, arising out of or in connection with the use or performance of the software or the copyrighted document.

Nokia Mobile Phones Ltd retains the right to make changes to this specification at any time without notice.

Licence:

You need a license from Nokia to use this document. The license terms will determine your rights and obligations. Without the prior written permission of Nokia, reproduction of the copyrighted document to any other server or location for further reproduction or redistribution is expressly prohibited.

Copyright © 2001. Nokia Mobile phones. All rights reserved.

1. INTRODUCTION

Java™ language is a general-purpose, concurrent, class-based, object-oriented programming language. It is designed to be simple enough that many programmers can achieve fluency in the language. Java is related to C and C++ but is organized rather differently, with a number of aspects of C and C++ omitted and a few ideas from other languages included.

The PersonalJava™ Application Environment (PJAE) is a Java application environment that executes software written in the Java programming language. The PJAE addresses the software needs of networked applications running on personal consumer devices such as set-top boxes and smart phones rather than desktop computers.

The PersonalJava Application Environment with JavaPhone™ 1.0 APIs is included in the Nokia 9210 Communicator.

The Nokia 9210 Communicator has the following features:

- Both PDA and CMT sides
- PersonalJava implemented only to the PDA side, with JavaPhone access also to the CMT side
- PDA screen properties:
  Color depth 12-bits, which is 4096 colors at a time
  PDA screen resolution 640x200
  No pointing device, different UI design (Virtual Cursor API)
  Common Button Area (CBA) on the right-hand side of the screen with four soft buttons
- A QWERTY keyboard with a menu key
- GSM data transports: HSCSD 2+2 / 3+1 / 1+1 with either 9600 bps or 14400 bps channel coding, SMS service with about 300 bps.
- Nokia 9210 Communicator hardware: 16 MB memory card in the sales package

2. JAVA IMPLEMENTATION ON THE NOKIA 9210 COMMUNICATOR

The Nokia 9210 Communicator is the first Java-capable Nokia phone. At a general level, the objectives of using Java in the Nokia 9210 Communicator are:

- To start fulfilling the promises of Java language: ease of programming, portability and safety. A simpler and safer language means easier programming, less errors, less testing and less debugging.
- Applet support: The Nokia 9210 Communicator's web browser doesn't supports applets, but appletviewer application is included. (An applet is a program written in the Java

Copyright © 2001. Nokia Mobile phones. All rights reserved.

programming language that can be included in an HTML page, much in the same way as an image is included.)

- Getting more 3rd party software: the Nokia 9210 Communicator has enough power for pJava applications, at least for games and utilities.

- Downloadable applets and applications: Java is a safe technology for it. A key feature of the current PC Internet industry is downloading the applications free of charge from the web.

- JavaPhone implementation: JavaPhone API is a vertical extension to the PersonalJava platform developed through an open process by Sun Microsystems in collaboration with other key expert groups in the telecommunications engineering community. JavaPhone APIs can only be used in applications, not in applets because of security reasons.

Nokia 9210 Communicator Java implementation is based on Sun Personal Java reference implementation (1.1.1 spec). It includes all optional functionality except printing. Java applications and applets can be run on both MARM (devices) and WINS (PC emulator). Customisation is needed only for the "look and feel" of the DFRDs (Device Family Reference Design).

3. DEVELOPMENT GUIDELINES

Developers can create add-on applications for the Nokia 9210 Communicator by registering on the Forum Nokia developer site and ordering "Nokia 9210 SDK for the Symbian Platform" on the Forum Nokia web site (www.forum.nokia.com). Both C++ and Java SDK are included on the CD-ROM.

The Java SDK contains an emulator for Windows NT4/95/98 (Win2000 not supported), documentation with examples and tools and additional tools to support native method development using JNI. Java compiler/IDE is not included in the SDK. One must download free JDKTM 1.1.8 + Personal Java compatibility classes from the Sun web site (java.sun.com/products/personaljava). The PersonalJava compatibility classes are additional classes available in the PersonalJava API which are not available in the corresponding Java Development Kit. For JNI development Microsoft Visual C++TM, v.6 is also needed.

The SDK documentation contains a Quick Start tutorial for creating a simple Hello World application and detailed information about developing Java applications for the Crystal platform. The Crystal platform is one of the Symbian's three EPOC DFRDs and it is designed for communicator-like devices.

3.1 User Interface Principles

The Nokia 9210 Style Guide is included in the SDK. It contains details on how the Crystal platform looks in the device. Certain aspects should be considered when developing applications for the Nokia 9210 Communicator. The interactive elements in the Crystal Platform consist of four command buttons, a menu and a PDA keyboard. No real pointing device or touch screen exist, but it's possible to use Pointer instead. The UI of the cellular telephone cannot be programmed.

Copyright © 2001. Nokia Mobile phones. All rights reserved.

Screen size (resolution 640 x 200) should be taken into account. The developer should also notice that certain areas in the display make use of the screen. Figure 1 shows a typical screenshot of the Nokia 9210 Communicator display. The Indicator and Command Button Areas have two different sizes or they can be hidden.

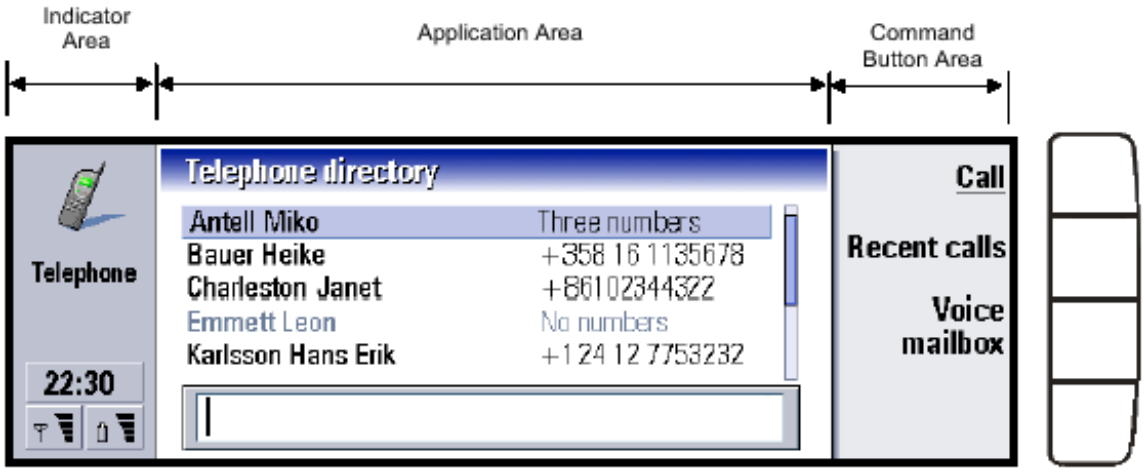


FIGURE 1. Typical view of the Nokia 9210 Communicator screen

3.2 Abstract Window Toolkit (AWT) interfaces

The Java programming language provides a class library called the Abstract Window Toolkit (AWT) that contains a number of common graphical widgets. You can add these widgets to your display area and position them with a layout manager.

A typical UI designed for a PC screen may not be efficient enough in the screen of the Nokia 9210 Communicator. It is recommended that application developers take advantage of the Crystal platform's UI features. Certain APIs for Nokia 9210 Communicator specific display features are included:

- Command Button Area API
- Status Pane API
- Virtual Cursor API

3.3 Command Button Area API

The Command Button Area is the recommended method of allowing the user to choose commands simply and rapidly. Crystal provides a class in the com.symbian.epoc.awt package that allows Java programs to create Command Button Areas and receive events from them. The API can be used by both applets and applications.

Copyright © 2001. Nokia Mobile phones. All rights reserved.

3.4 Status Pane API

The Status Pane API allows you to access Crystal's Status Pane (sometimes referred to as the Indicator Area). The Status Pane is dedicated to providing different kinds of important information such as connection, battery and field strength indicators. The API allows you to set the style (size) of the Status Pane. There are two styles for the Status Pane: the wide area, which has 92 x 200 pixels, and the narrow one, which has 32 x 200 pixels. The wide area also has a place for time, the application's icon and name.

Crystal provides a class in the com.symbian.epoc.awt package that allows Java programs to control the Status Pane. The API can be used by both applets and applications.

3.5 Virtual Cursor API

The Virtual Cursor (VC, i.e. Pointer) is a useful UI extension providing users with a software pointer in the absence of a hardware pointer (e.g. mouse pointer). The Pointer is useful in applications that explicitly require pointer events. For example, image maps on a web browser requiring pointer events can be handled by the Pointer: the VC is enabled and its actions are received as normal pointer events by the browser application.

Crystal provides a class in the com.symbian.epoc.awt package that allows Java programs to control the Pointer and receive pointer events from it. The API can be used by both applets and applications.

More information about Symbian classes can be found after SDK installation in the documentation: Crystal V6 Edition for Java -> Crystal Java API Reference -> package com.symbian.epoc.awt and C:\Symbian\6.0\NokiaJava\erj\utilities\Awt\JavaDoc\index.html.

4. SHORT CODE EXAMPLE

In this chapter we have a short code example. The example code gives basic knowledge about developing Java applications for the Nokia 9210 Communicator (and Crystal platform). The application is a simple "Hello World!"-application. It is based on the CBADemo application included with the SDK CD-ROM. This documentation presumes that the SDK is installed in the default folder c:\Symbian\6.0\NokiaJava\.

Copyright © 2001. Nokia Mobile phones. All rights reserved.

Figure 2 gives a general overview on how to develop applications for the Nokia 9210 Communicator. Basically it doesn't differ from writing Java code. Tools for running the application in the emulator or in the target device are included in the SDK.

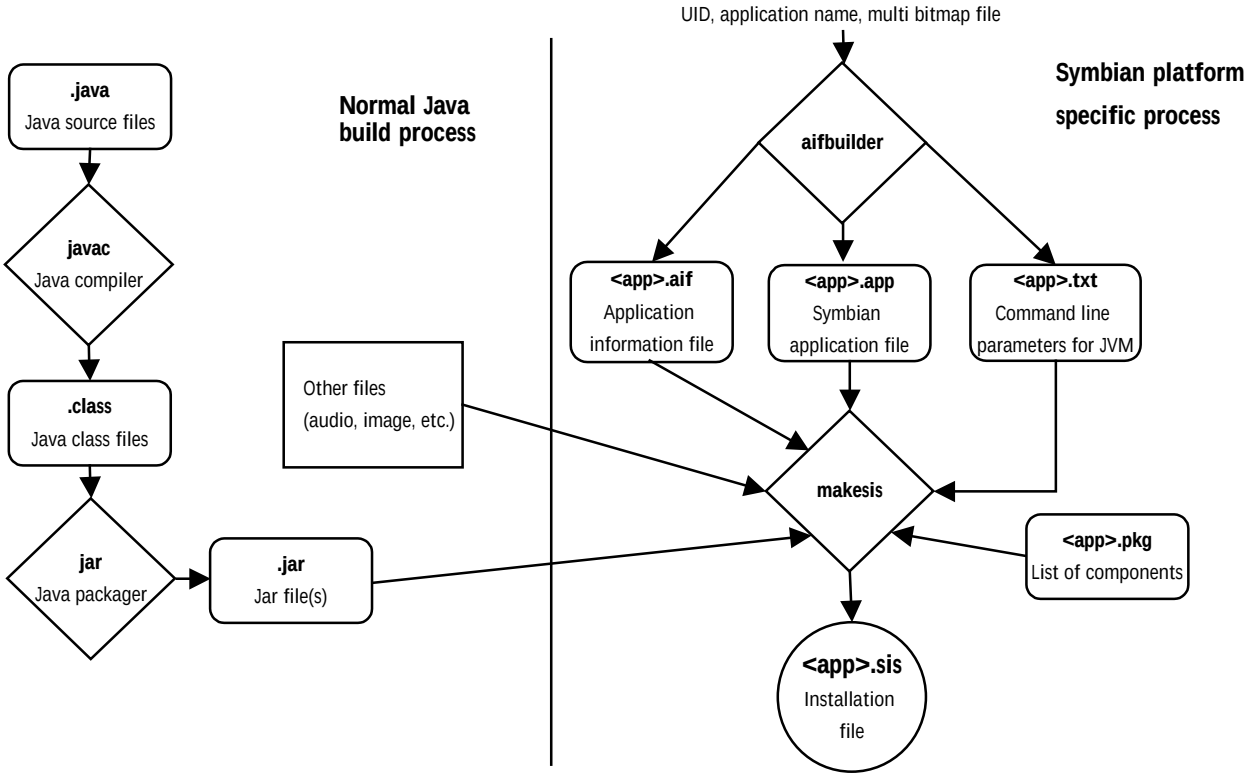


FIGURE 2. Java application creation process for the Nokia 9210 Communicator

Writing and compiling Java source code is as usual; code files can be archived using the jar command (for example jar cf MyApp.jar *.class). Also .zip format can be used. Add-on applications can be made visible in the Extras screen in the Nokia 9210 Communicator. The applications should be installed on the !:\system\apps\<appname>\, where ! is the letter of the imaginary drive (meaning either the Communicator's memory or Memory Card). When installing the end-user has the opportunity to choose the destination drive of the installation.

Now in this small application we are using Symbian's own classes. The path of these classes should be added to the classpath variable. This can be done by editing the classpath variable in the autoexec.bat (Windows 98) or in Control Panel -> System -> Environment (Windows NT). For example in Windows 98 the classpath variable could be as follows:

```
classpath=.;c:\jdk1.1.8\lib\classes.zip;c:\jdk1.1.8\lib\classes\class
es.zip; c:\jdk1.1.8\lib\classes\cawt.jar
```

Copyright © 2001. Nokia Mobile phones. All rights reserved.

where the c:\jdk1.1.8\lib\classes\ folder contains Symbian classes. More information about classpath issues with the emulator can be found in the SDK documentation:

Crystal V6 Edition for Java -> Java Developer's Guide for Crystal -> Java Development -> The default classpath.

4.1 The Source Code

Below is the source code of the amazing Hello World application. The code should be copied to the file Helloworld.java and compiled with javac.exe.

```
import  com.symbian.devnet.crystal.awt.*;
import  com.symbian.epoc.awt.*;
import  java.awt.*;

class Helloworld extends CFrame implements CBAListener
{
  private static GridPanel panel = new GridPanel();
  private CBAHandler cba;

  static final String TITLE = "Hello World";
  final static int CLOSE_BUTTON = EikCommandButtonGroup.BUTTON4;

  public static void main( String args[] )
  {
     new Helloworld();
  }

  public Helloworld()
  {
     setTitle(TITLE);
     cba = new CBAHandler( this );
     cba.setText( CLOSE_BUTTON, "Close" );
     cba.activate();

     Font font = new Font("SansSerif",0,30);
     Label label = new Label("Hello World!!!");
     label.setFont(font);

     panel.add(label,0,1);
     add(panel);

     show();
  }

  public void cbaActionPerformed( CBAEvent e )
  {
     switch ( e.getID() )
     {
        case CLOSE_BUTTON:
        shutDown();
        break;
     }
  }
}
```

4.2 Symbian Platform Specific Processes

Certain procedures must be followed to make the Hello World application visible in the emulator and the target machine. One must use the tool applications included with the SDK. The AIF Builder tool writes certain files needed for the Symbian operating system and the makesis.exe

Copyright © 2001. Nokia Mobile phones. All rights reserved.

command makes an installation package of the files. During these processes the application also gets its own unique identifier (UID) and icons.

Every Symbian EPOC application should have its own UID. This allows the operating system to distinguish files associated with that application from files with other applications. A UID is a 32-bit number, which you get as you need from Symbian (More info can be found on www.symbiandevnet.com).

According to Figure 2, the AIF Builder tool needs a UID, an application name and bitmap pictures and as a result it gives us:

1. An .aif file, Application Information File
2. An .app file, Application file, containing the UID
3. A .txt file, command line parameters for pjava.exe (a JVM executable file)
4. An .mbm file, multi bitmap file for icons

The first step is to start the AIF Builder tool from Start Menu -> Programs -> Symbian 6.0 SDKs -> Development Tools -> AIF Builder. Then choose File -> New from the menu.

Files for an application must be located in a directory \System\Apps\app_name\, on any drive, where app_name is the application's name. When working with the Java version of the emulator you should create a suitable directory manually yourself. Any name may be assigned to *app_name* — there is no requirement that it must have the same name as the class or jar file. In Figure 3 can be seen the starting screen after creating a new AIF Builder file.

You can see the application's details in Figure 3. The programming language is chosen to be Java, the Application Name is Helloworld, the UID is set to be 0x10001122 (in hexadecimal form; it could also be in decimal form) and finally the Java Command Line Text "-Dcom.symbian.appName="Hello World" Helloworld". The application name is set in the command line text and the Helloworld word in the end tells the application's main class name to the pjava.exe. There could also be other definitions, for example additional classes to the classpath variable, a working directory, etc.

Copyright © 2001. Nokia Mobile phones. All rights reserved.

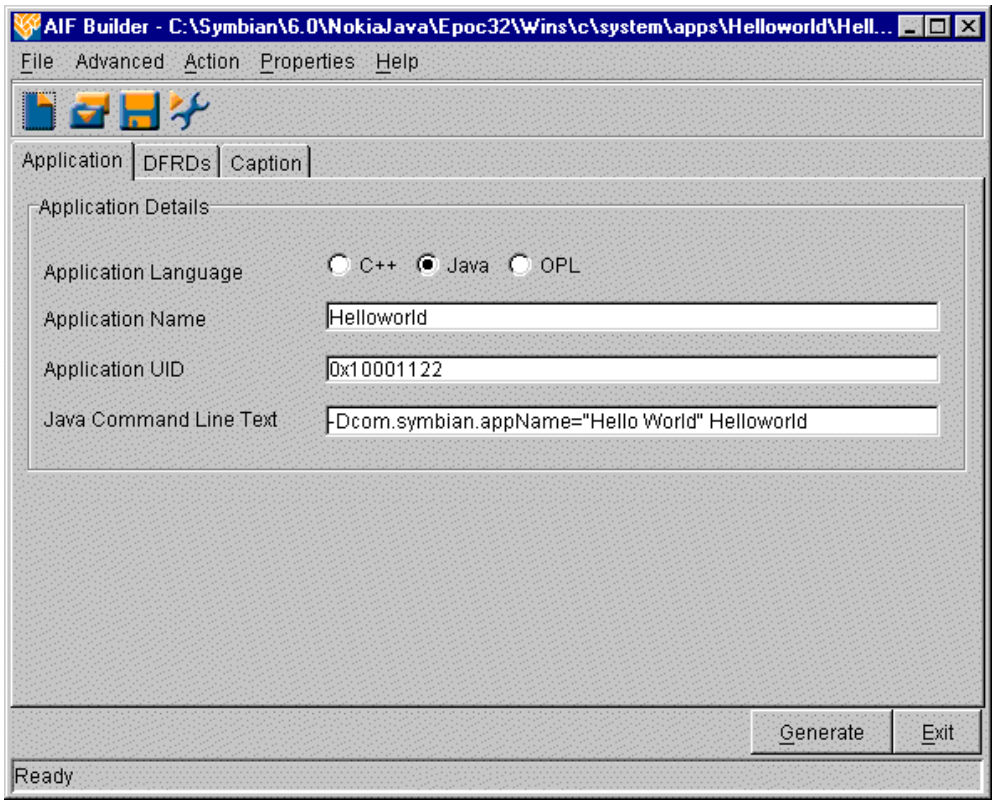


FIGURE 3. AIF Builder tool's starting screen

The next step is to generate your own personal icons for the Helloworld application. The AIF Builder tool can be used as an icon editor. Press the DFRDs tab and then press the Edit button. Pressing the button opens an editor view which is shown in Figure 4. You can draw the icons for two resolutions: 25x20 and 64x50. Then you can save the multi bitmap file and close the icon drawing application. It might be a good idea to save the .mbm picture to the same folder as the AIF Builder tool's .aifb file (look at the picture below). The tool also generates a folder named Helloworld_BMPs and writes individual .bmp files of the icons and their mask files. You can read more about this subject in the Professional Symbian Programming book.

Copyright © 2001. Nokia Mobile phones. All rights reserved.

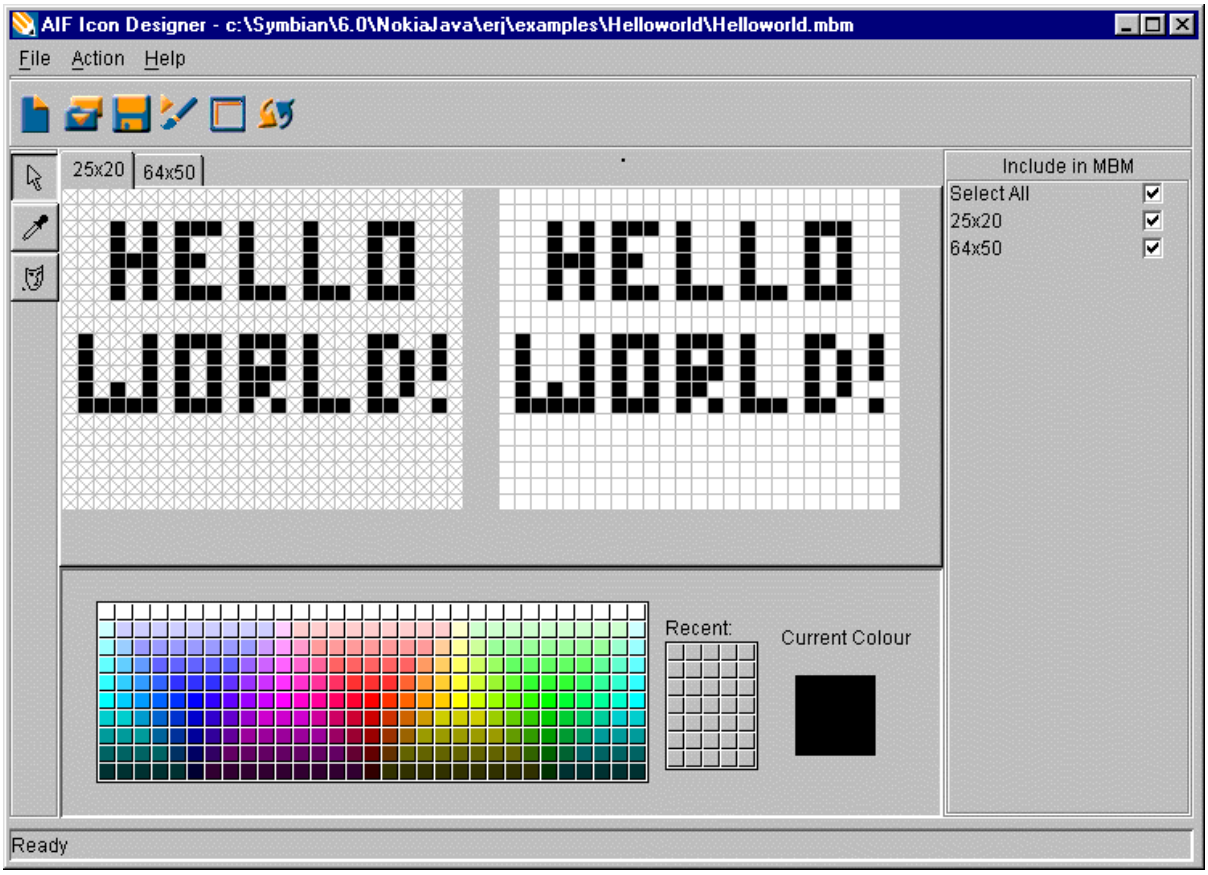


FIGURE 4. Picture of AIF Icon Designer

The DFRDs screen is shown in Figure 5. After saving the icon file you can write the file name with the correct path to the AIF Builder window (Name of MBM textbox). Remember to uncheck the Quartz box in the 'DFRDs to Generate for' box, as we are writing this application for the Crystal platform only. You should also write the path where the AIF Builder tool generates the files. The working directory c:\Symbian\6.0\NokiaJava\erj\examples\Helloworld\ might be a good place for them.

Finally you are asked for the path for the output directory for temporary files. These files are really not needed after application specific files are generated, so it doesn't matter where you put them. Generally c:\temp or c:\windows\temp is used for temporary files and it can be used also here.

The last screen of the three screens in the AIF Builder tool is the Caption screen. It is used for setting captions in different languages. 23 languages are supported. You can get more information about this issue from http://www.symbiandevnet.com.

Copyright © 2001. Nokia Mobile phones. All rights reserved.

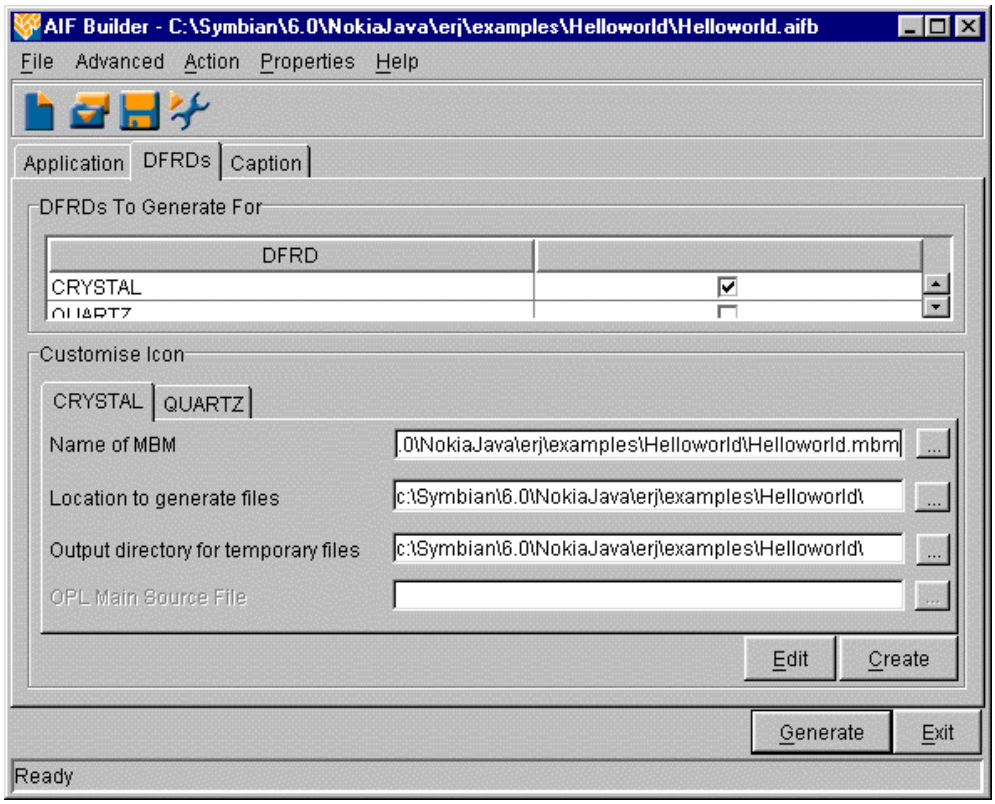
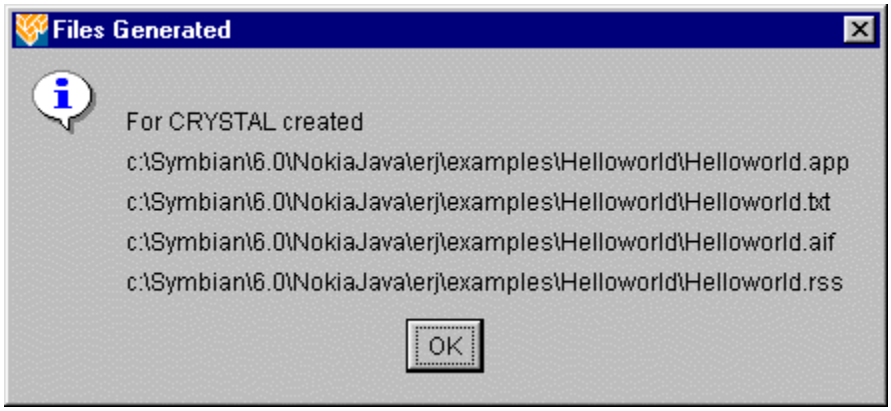


FIGURE 5. AIF Builder tool's DFRDs screen

After finishing filling these three pages in the AIF Builder tool you are ready to generate the files for the application. Just press the Generate button in the right bottom corner. If everything is set properly you should get a window like that shown below:



In this case the SDK is not installed in the default directory. Here the SDK is installed on the own drive made by using the subst command, but this is not necessary. In some cases it might be clearer to have all the SDK files (and nothing else) in the same drive.

Anyway, now you should have at least the following files in your working folder; the files displayed in bold should be installed for running the application:

Copyright © 2001. Nokia Mobile phones. All rights reserved.

- Helloworld.java — source code
- **Helloworld.class** — compiled from source file
- **Helloworld.txt** — command line parameters for the JVM
- Helloworld.aifb — AIF Builder tool file
- **Helloworld.aif** — Application information file of the Helloworld application
- **Helloworld.mbm** — Multi bitmap file
- **Helloworld.app** — Application file
- Helloworld.lst — Text file which lists the content of the Helloworld_BMPs folder
- Helloworld_BMPs (folder) — Folder containing the .bmp files of the .mbm file

Now you have everything for running the application in the emulator! There are several possible ways of launching an application in the emulator. Generally, to make an application visible in the Extras view, one should copy application files to the c:\Symbian\6.0\NokiaJava\epoc32\wins\c\System\apps\<appname> folder, where <appname> is the name of the application. Then just open the .app file of the application in the File Manager. This also works similarly with the target device; in that case the folder is \System\apps\<appname> (either in the device's memory or memory card). One way is to make a .bat file for launching JVM (pjava.exe) with command line parameters. (The example applications with the SDK can be launched by running the included .bat files.)

After a successful launch of the applications you should get a screen similar to the one in Figure. 6. The application name Hello World (set in the command line parameters in the AIF Builder) is shown in the blue title bar. The Close button is set, according to the Style Guide's instructions, to the fourth button. You can also look at the menu commands by pressing the F1 button; these commands are set automatically.

FIGURE 6. Helloworld application in the emulator

4.2.1 Making an installation package for the target device

For a convenient installation procedure one should make an installable package of the application. The SDK includes the necessary tool, makesis.exe, for making a .sis file, Symbian's own installation file. The makesis tool needs the .pkg file as an input, where for example the

Copyright © 2001. Nokia Mobile phones. All rights reserved.

application files to be copied to the right folder of the device are listed. The file Helloworld.pkg with added row numbers is listed below:

```
1 &EN
2 #{"Helloworld"},(0x10001122),1,0,0,TYPE=SISAPP
3 "Helloworld.aif"-"!:\system\apps\Helloworld\Helloworld.aif"
4 "Helloworld.txt"-"!:\system\apps\Helloworld\Helloworld.txt"
5 "Helloworld.mbm"-"!:\system\apps\Helloworld\Helloworld.mbm"
6 "Helloworld.class"-"!:\system\apps\Helloworld\Helloworld.class"
7 "Helloworld.app"-"!:\system\apps\Helloworld\Helloworld.app"
```

In the first row is specified the language variants which are supported; in this case the only supported language is English. In row 2 is set the name of the application, UID and major and minor build numbers and also the installation type is set. The UID number in the .pkg file must be same as that set in the AIF Builder file. More information can be found in the SDK documentation (check "Application Installation Guide"). Lines 3-7 indicate which files are copied and to where.

In this case the command for making the .sis file would be:

c:\Symbian\6.0\NokiaJava\erj\examples\Helloworld\>makesis Helloworld.pkg

and as an output one would get the file Helloworld.sis. This installation file contains all files listed in the .pkg file. The .sis file can be copied to the device by using the infrared port or cable connection (or applications can be even downloaded from the Internet). The Helloworld application can now be installed by opening the .sis file in the device. After installing one should find the self-made Helloworld icon in the Extras screen. The application can now be launched by pressing the Open button.

For running this application one should remember that certain Java classes should be copied to the device, if they are not present. The file \system\Java\ext\cawt.jar should be present. This packet contains Symbian's own Java classes, which are used in all the SDK example applications. Developers don't have to use the classes in cawt.jar, but there is useful classes for making use of basic UI components like Menus and Command Buttons.

5. FURTHER INFORMATION

1. Nokia 9210 Style Guide (included in the SDK CD-ROM)
2. Java Developer's Guide for Crystal (included in the SDK CD-ROM)
3. Forum Nokia Web Site http://www.forum.nokia.com
4. Java Web Site http://java.sun.com
5. Symbian Developer Net http://www.symbiandevnet.com
6. Professional Symbian Programming, Martin Tasker et al., Wrox Press Ltd., 2000
7. Example applications included in the SDK (C:\Symbian\6.0\NokiaJava\erj\examples)
8. Personal Java Web Site http://java.sun.com/products/personaljava/spec-1-1-1/index.html

Copyright © 2001. Nokia Mobile phones. All rights reserved.

LIST OF TERMS AND ABBREVIATIONS

Term or abbreviation	Description
API	Application Programming Interface
AWT	Abstract Window Toolkit
CBA	Command Button Area
CMT	Cellular Mobile Telephone
DFRD	Device Family Reference Design
HSCSD	High Speed Circuit Switched Data
IDE	Integrated Development Environment
JAR	Java Archive
JDK	Java Development Kit
JNI	Java Native Interface
JVM	Java Virtual Machine
MBM	Multi Bitmap file
PDA	Personal Data Assistant
PJAE	PersonalJava Application Environment
pJava	PJAE, PersonalJava™ Technology
SDK	Software Development Kit
UID	Unique Identification code

Copyright © 2001. Nokia Mobile phones. All rights reserved.